EE:211

Computational Techniques in Electrical Engineering

Lab#2

Polynomial Interpolation using Matlab-I

1. **Use of Matlab command polyfit ( ) to implement polynomial interpolation and use of polyval ( ) to evaluate the polynomial.**

To see what the function **polyfit( )** does, type this at Matlab command window:

>> help polyfit

And it gives the following definition:

P = POLYFIT(X,Y,N) finds the coefficients of a polynomial P(X) of degree N that fits the data Y best in a least-squares sense. P is a row vector of length N+1 containing the polynomial coefficients in descending powers, P(1)*X^N + P(2)*X^(N-1) +...+ P(N)*X + P(N+1).

Let's implement the Example 4.1.1 on page: 119 of the textbook, which implement the linear interpolation for the data points, x= [1, 4], y= [1, 2]. If we simplify the equation (4.2) then we get the following polynomial:

$P_1(x)=0.333x+0.667$ -------- **equation(1)**

To determine the above polynomial representation uses the following Matlab code:

```
>> x= [1, 4], y= [1, 2]

>> p1=polyfit(x,y,1)

   p1 = 0.3333    0.6667
```

This give the coefficient for linear interpolation polynomial (use n=1) as given in equation (1), Suppose we want to evaluate this polynomial as given by equation (1) at x=2, i.e. $P_1(2)$= 1.3333. We can use the Matlab command **polyval ( )** to evaluate the polynomial at any point x. For the p1 coefficient calculated above use the following code to evaluate the above polynomial at x=2.

```
>> y=polyval(p1,2)  y =1.3333
```

Consider the Example 4.1.3 on page:121, which determine the quadratic polynomial for the data points x=[0 1 2] and y=[-1 -1 7]. The simplified equation (4.9) is given below:

$P_2(x)=4x^2-4x-1$ -------- **equation(2)**

To find the $P_2(x)$ representation (in terms of coefficient) use the following code:

```
>> x=[ 0 1 2] , y=[-1 -1 7]

>> p2=polyfit(x,y,2)

p2 = 4    -4    -1
```

2. Constructing the Lagrange Interpolating Polynomial using Matlab

1. In order to construct the Lagrange Coefficients for the Lagrange Polynomial in MATLAB, we can use the built-in function **poly**, which constructs a polynomial with given roots. Enter the following to construct a polynomial with roots 1 and 2 for example:

```
» poly([1 2])
ans =
1  -3  2
```

Thus, this is the polynomial (x-1)(x-2) = x^2 -3x +2 which has roots 1 and 2.

2. Consider the Example 4.1.3, page 121 which construct Lagrange interpolating polynomial of degree two (quadratic) approximating the data points [(0,-1),(1,-1),(2,7)]

Consider the Lagrange basis function $L_0(x)$ given as

$$L_0(x) = \frac{(x-1)(\,x-2)}{(0-1)(0-2)}$$

We can see that we need the numerator to be the polynomial with roots $x_1 = 1$, and $x_2 = 2$, i.e. (*x* -1)(*x*-2) The denominator is the constant $(x_0 - x_1)*(x_0 - x_2) = (0-1)*(0-2) = 2$.

Assuming that we want to store the Lagrange coefficient polynomials in the 3x3 array (matrix) L (with the 1st row being the coefficients for $L_0(x)$, the 2nd row being the coefficients for $L_1(x)$, and the third row being the coefficients for $L_2(x)$), we proceed as follows:

```
>> L(1,:)= poly([1  2])/((0 - 1)*(0 - 2))

L = 0.5000   -1.5000    1.0000

>> L(2,:)= poly([0  2])/((1 - 0)*(1 - 2))

L = 0.5000   -1.5000    1.0000

    -1.0000    2.0000       0
```

```
>> L(3,:)= poly([0  1])/((2 - 0)*(2 - 1))

    L =

     0.5000   -1.5000    1.0000

   -1.0000    2.0000         0

    0.5000   -0.5000         0
```

The final Lagrange polynomial is: $P_2(x)= y_0 * L_0(x)+ y_1 * L_1(x)+y_2 * L_2(x)$. We compute the $P_2(x)$ as:

```
>> P = (-1)*L(1,:) + (-1)*L(2,:) + (7)*L(3,:)

P = 4    -4    -1
```

This has the same coefficient as equation (2) above.

To evaluate the polynomial at 2 we use:

```
>> polyval(P,2)

ans =7
```